Available online at www.sciencedirect.com
ScienceDirect
Discrete Applied Mathematics 156 (2008) 3345–3361
DISCRETE APPLIED MATHEMATICS
www.elsevier.com/locate/dam

Closest 4-leaf power is fixed-parameter tractable☆

Michael Dom*, Jiong Guo, Falk Hüffner, Rolf Niedermeier

Institut für Informatik, Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, D-07743 Jena, Germany

Received 3 August 2006; received in revised form 18 December 2007; accepted 9 January 2008
Available online 4 March 2008

Abstract

The NP-complete CLOSEST 4-LEAF POWER problem asks, given an undirected graph, whether it can be modified by at most r edge insertions or deletions such that it becomes a 4-leaf power. Herein, a 4-leaf power is a graph that can be constructed by considering an unrooted tree—the 4-leaf root—with leaves one-to-one labeled by the graph vertices, where we connect two graph vertices by an edge iff their corresponding leaves are at distance at most 4 in the tree. Complementing previous work on CLOSEST 2-LEAF POWER and CLOSEST 3-LEAF POWER, we give the first algorithmic result for CLOSEST 4-LEAF POWER, showing that CLOSEST 4-LEAF POWER is fixed-parameter tractable with respect to the parameter r.

Keywords: Fixed-parameter tractability; Graph algorithm; Graph modification; Graph power; Leaf power; Forbidden subgraph characterization

1. Introduction

Graph powers form a classical concept in graph theory, and the rich literature dates back to the sixties of the previous century (see [5, Section 10.6] and [27] for surveys). The *k-power* of an undirected graph $G = (V, E)$ is the undirected graph $G^k = (V, E')$ with $(u, v) \in E'$ iff there is a path of length at most k between u and v in G. We say G is the *k-root* of G^k. It is NP-complete to decide whether a given graph is a 2-power (*square*) [30]. By way of contrast, one can decide in linear time whether a graph G is a k-power of a tree for any fixed k [9,26,29], and one can also find an integer k and a tree T in linear time such that $G = T^k$ [9].

In this paper we concentrate on certain practically motivated variants of tree powers. Whereas Kearney and Corneil [22] studied the problem where every tree node one-to-one corresponds to a graph vertex, Nishimura, Ragde, and Thilikos [33] introduced the notion of *leaf powers* where the tree leaves exclusively stand in one-to-one correspondence with the graph vertices. In addition, Lin, Kearney, and Jiang [28], Chen, Jiang, and Lin [11], and Chen and Tsukiji [12] examined the variant of leaf powers where all inner nodes of the root tree have degree at

☆ Research supported by the Deutsche Forschungsgemeinschaft (DFG), Emmy Noether research group PIAF (fixed-parameter algorithms), NI 369/4. A preliminary version of this paper appeared under the title "Extending the Tractability Border for Closest Leaf Powers" in the Proceedings of the 31st International Workshop on Graph-Theoretic Concepts in Computer Science, WG '05, in: LNCS, vol. 3787, Springer, 2005, pp. 397–408.
* Corresponding author.
E-mail addresses: dom@minet.uni-jena.de (M. Dom), guo@minet.uni-jena.de (J. Guo), hueffner@minet.uni-jena.de (F. Hüffner), niedermr@minet.uni-jena.de (R. Niedermeier).

0166-218X/$ - see front matter
doi:10.1016/j.dam.2008.01.007

least three. Both problems find applications in computational evolutionary biology [11,28,33]. The corresponding recognition problems are called k-LEAF POWER [33] and k-PHYLOGENETIC ROOT [28], respectively.[1] k-LEAF POWER is solvable in linear time for $k \leq 5$ [4,6,8], and k-PHYLOGENETIC ROOT is solvable in polynomial time for $k \leq 4$ [33,28]. The complexities of both recognition problems are open for $k \geq 6$ and $k \geq 5$, respectively, although it is known that every so-called *strictly chordal* graph is a k-leaf power for every k [24] (see also [3] for similar results on further graph classes), and 5-PHYLOGENETIC ROOT can be solved in cubic time on strictly chordal graphs [23].

Several groups of researchers [11,22,28] advocated the consideration of a more relaxed or "approximate" version of the graph power recognition problem: Now, look for roots whose powers are *close* to the input graphs, thus turning the focus of study to the corresponding *graph modification* problems. Kearney and Corneil [22] were the first to formulate this problem setting when introducing the CLOSEST k-TREE POWER problem. In this "error correction setting" the question is whether a given graph can be modified by adding or deleting at most r edges such that the resulting graph has a k-tree root. This problem turns out to be NP-complete for $k \geq 2$ [22,21,14]. One also obtains NP-completeness for the corresponding problems CLOSEST k-LEAF POWER [25,13] and CLOSEST k-PHYLOGENETIC ROOT [11,36].

All nontrivial ($k \geq 2$) "approximate recognition" problems in our context turn out to be NP-complete [2,11,13,14, 21,22,25,35,36]. Hence, the pressing quest is to also show positive algorithmic tractability results such as polynomial-time approximation or nontrivial (exponential-time) exact algorithms. For the most simple version of CLOSEST k-LEAF POWER, $k = 2$, intricate polynomial-time constant-factor approximation algorithms have been developed [2,10,1,37]. After a series of improvements, the best known polynomial-time approximation is by a factor of 2.5 [1, 37].[2] Moreover, it is fairly easy to show that for $k = 2$ the problem is fixed-parameter tractable with respect to the parameter r denoting the number of allowed edge modifications [19]. In particular, efficient polynomial-time data reduction rules have been proposed, which yield so-called *problem kernels* consisting of only $O(r)$ vertices [17,20]. At least with respect to these fixed-parameter tractability results, the success is surely due to the fact that there is a very simple characterization by a forbidden subgraph: a graph is a 2-leaf power iff it contains no induced 3-vertex subgraph forming a path. Observe that also the recognition problem for 2-leaf powers is solvable in linear time by just checking whether the given graph is a disjoint union of cliques. By way of contrast, the recognition problem for 3-leaf and 4-leaf powers is much harder [4,6,33]. At first sight, this lowers the hope for obtaining positive algorithmic results for CLOSEST k-LEAF POWER for $k = 3, 4$. The key idea we put forward here and in a companion paper [13] is to again develop and employ forbidden subgraph characterizations of the respective graph classes. Unlike for 2-leaf powers, these characterizations are not so obvious. In [13], we described a forbidden subgraph characterization for 3-leaf powers, consisting of five graphs of small size. Here, we employ a forbidden subgraph characterization for 4-leaf powers—it requires numerous forbidden subgraphs.

Let us discuss the algorithmic use of these forbidden subgraph characterizations. First, both characterizations immediately imply polynomial-time recognition algorithms for 3- and 4-leaf powers that are conceptually simpler than those in [4,6,8,33]. However, our algorithms are of purely theoretical interest because the running times of these straightforward algorithms are much worse than that of the known cubic-time algorithms from [33] and linear time algorithms from [4,6,8]. More important, the characterizations open up the way to the first tractability results for the harder problems CLOSEST k-LEAF POWER for $k = 3, 4$. Using the forbidden subgraphs for 3-leaf powers, in [13] we showed that CLOSEST 3-LEAF POWER is fixed-parameter tractable with respect to the parameter "number r of edge modifications." Due to the significantly increased combinatorial complexity of 4-leaf powers (with numerous forbidden subgraphs instead of only a handful), analogous results for CLOSEST 4-LEAF POWER remained open in [13]. We close this gap here. We show that CLOSEST 4-LEAF POWER can be solved in polynomial time for $r = O(\log n / \log \log n)$; that is, it is fixed-parameter tractable with respect to the parameter r. Moreover, the variants of CLOSEST 4-LEAF POWER where only edge insertions or only edge deletions are allowed are fixed-parameter tractable as well. On the way to our main result (Section 4), we develop a "compressed form" of a forbidden subgraph characterization of 4-leaf powers (Section 3) that has been developed—independently and by different means—by

[1] Both problems k-LEAF POWER and k-PHYLOGENETIC ROOT ask whether a given graph is a leaf power resp. a phylogenetic power. We find it more natural to use the term *power* instead of the term *root* here, although we used the term *root* in the conference version of our previous considerations concerning the case $k = 3$ [13].

[2] Note that in the various papers (partially not referring to each other) CLOSEST 2-LEAF POWER appears under various names such as CLUSTER EDITING [35] and CORRELATION CLUSTERING [1,2,10,37].

Rautenbach [34]. Since we aim at algorithmic tractability results for CLOSEST 4-LEAF POWER, we employ a "more constructive" approach.

2. Preliminaries

We consider only undirected graphs $G = (V, E)$ with $n := |V|$ and $m := |E|$. Edges are denoted as tuples (u, v), ignoring any ordering. For a graph $G = (V, E)$ and $u, v \in V$, let $d_G(u, v)$ denote the length of a shortest path between u and v in G. With $E(G)$, we denote the edge set E of a graph G. We call a graph $G' = (V', E')$ an *induced subgraph* of $G = (V, E)$ and denote G' with $G[V']$ if $V' \subseteq V$ and $E' = \{(u, v) \mid u, v \in V' \text{ and } (u, v) \in E\}$. For a nonempty collection of graphs $\mathcal{G}$, a graph is said to be *$\mathcal{G}$-free* if it does not contain any graph in $\mathcal{G}$ as an induced subgraph. A cycle with n vertices is denoted as C_n. An edge between two vertices of a cycle that is not part of the cycle is called *chord*. An induced cycle of length at least four is called *hole*—note that a hole is chordless. A *chordal graph* then is a hole-free graph. Let a *minimum edge cut*, denoted by MINCUT (G, V_1, V_2), be a minimum weight set of edges in $G = (V, E)$ that disconnects all vertices in $V_1 \subseteq V$ from those in $V_2 \subseteq V$. We say that a set is *maximal* with respect to some property if it is not a proper subset of another set with that property. For two sets A and B, $A \triangle B$ denotes the *symmetric difference* $(A \setminus B) \cup (B \setminus A)$.

Definition 2.1 (*[33]*). Consider an unrooted tree T with leaves one-to-one labeled by the elements of a set V. The *k-leaf power* of T is a graph, denoted T^k, with $T^k := (V, E)$, where $E := \{(u, v) \mid u, v \in V \text{ and } d_T(u, v) \leq k\}$. We call T a *k-leaf root* of T^k.

The k-LEAF POWER (LPk) problem then is to decide, given a graph G, whether there is a tree T such that $T^k = G$.

One may view the leaf power concept as a "Steiner extension" of the standard notion of tree powers [11,28]. The more general, *approximate version* of LPk we focus on in this work, called CLOSEST k-LEAF POWER (CLPk), then reads as follows. Given a graph $G = (V, E)$ and a nonnegative integer r, is there a tree T such that T^k and G differ by at most r edges, that is, $|E(T^k) \triangle E(G)| \leq r$? CLP$k$ is NP-complete for $k \geq 2$ [25,13].

In this paper we also study two variations of CLPk referring to only one-sided errors: CLPk EDGE INSERTION only allows insertion of edges and CLPk EDGE DELETION only allows deletion of edges. CLPk EDGE DELETION is NP-complete for $k \geq 2$ [31,13], and CLPk EDGE INSERTION is NP-complete for $k \geq 3$ but trivially polynomial-time solvable for $k = 2$ [13].

A central technical tool within this work are critical cliques and critical clique graphs as Lin et al. [28] introduce them.

Definition 2.2. A *critical clique* of a graph G is a clique K where the vertices of K all have the same set of neighbors in $G \setminus K$, and K is maximal under this property. Consider a graph $G = (V, E)$. Let V_C be the collection of its critical cliques. Then the *critical clique graph* $\mathrm{CC}(G)$ is a graph (V_C, E_C) (we use the term *nodes* for its vertices) with

$$(K_i, K_j) \in E_C \iff \forall u \in K_i, v \in K_j : (u, v) \in E.$$

That is, the critical clique graph has the critical cliques as nodes, and two nodes are connected iff the corresponding critical cliques together form a larger clique.

See Fig. 1 for an illustration. Note that if G is chordal, then so is $\mathrm{CC}(G)$, since if $\mathrm{CC}(G)$ contained a hole, we could also find a hole in G by taking one arbitrary vertex from each critical clique on the cycle in $\mathrm{CC}(G)$. Given a chordal graph G, its critical clique graph $\mathrm{CC}(G)$ can be computed in $O(n + m)$ time [28].

Eventually, for technical reasons we also need the concept of a k-Steiner root.

Definition 2.3. Consider a graph $G = (V, E)$ and an arbitrary set of vertices A with $A \cap V = \emptyset$. An unrooted tree $T = (A \cup V, E')$ is called a *k-Steiner root* of G if $E = \{(u, v) \mid u, v \in V \text{ and } d_T(u, v) \leq k\}$.

Note that if $A = \emptyset$, then a k-Steiner root simply is a k-tree root. Similarly, if A is the set of inner nodes of T, then a k-Steiner root is the same as a k-leaf root. This means that the set of graphs that have k-Steiner roots is a superset of the set of graphs that have k-tree roots or k-leaf roots. The following lemma is easy to show (a similar statement was already made by Lin et al. [28]).

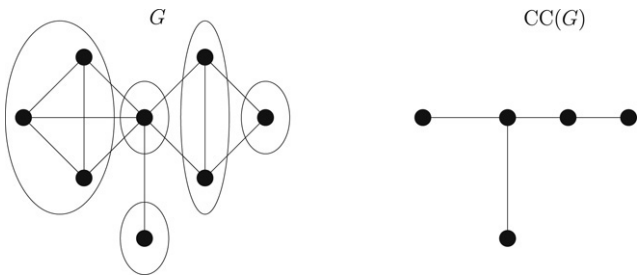

Fig. 1. A graph G (left) with critical cliques marked by ellipses, and its critical clique graph $\mathrm{CC}(G)$ (right).

Lemma 2.1. *A graph G has a k-leaf root iff $\mathrm{CC}(G)$ has a $(k-2)$-Steiner root.*

We show that CLP4 and both its edge insertion and edge deletion variants are *fixed-parameter tractable* (FPT) with respect to the parameter r. That is, we show that CLP4 can be solved in $f(r)\cdot n^{O(1)}$ time, where f is a computable function only depending on r, and n denotes the number of vertices of the input graph. More on fixed-parameter tractability and parameterized complexity can be found in the monographs [15,18,32].

3. Forbidden subgraph characterization of 4-leaf powers

In this section we give a characterization of 4-leaf powers using a set of eight forbidden induced subgraphs for the critical clique graphs of 4-leaf powers. This set can be extended to a larger set of forbidden subgraphs for the 4-leaf powers themselves by a simple iterative algorithm. Independently and by different proof techniques, Rautenbach [34] achieves the same results. Our approach, however, is tailored towards the algorithmic treatment following in the next section. The eight forbidden subgraphs for critical clique graphs of 4-leaf powers are shown in Fig. 2. Let $\mathcal{F} := \{F_1, F_2, \ldots, F_8\}$ as given there.

The main result of this section is as follows:

Theorem 3.1. *For a graph G, the following are equivalent:*

(1) *G is a 4-leaf power.*
(2) *G is chordal and its critical clique graph $\mathrm{CC}(G)$ is $\mathcal{F}$-free.*

The forbidden subgraph characterization of Theorem 3.1 refers to critical clique graphs. However, it directly implies a somewhat more extensive forbidden subgraph characterization for the original graphs.

Corollary 3.1. *All 4-leaf powers are chordal, and chordal graphs that are 4-leaf powers can be characterized by a finite set of forbidden subgraphs.*

Proof. Consider a graph G with its critical clique graph $\mathrm{CC}(G)$. Using Theorem 3.1, one forbidden subgraph from the set $\mathcal{F}$ corresponds to several subgraphs in G: If a graph $F_i \in \mathcal{F}$ is an induced subgraph of $\mathrm{CC}(G)$, then there is also an induced F_i in G. Moreover, if F_i contains a pair u, v of adjacent nodes with the same neighborhood, then one can find an induced F_i in G plus a vertex that is adjacent to exactly one of the critical cliques represented by u and v; otherwise, since each critical clique is maximal under the property of having the same neighborhood, there could not be two distinct nodes u and v in $\mathrm{CC}(G)$. This vertex, which we call a *distinguishing vertex*, may be adjacent to every combination of the critical cliques represented by the other nodes of F_i and, if there are more pairs of adjacent nodes with the same neighborhood in F_i, to every combination of the other distinguishing vertices added to F_i. By examining all these combinations and weeding out isomorphic and nonchordal graphs, we can construct the complete set of forbidden subgraphs for G. For instance, $F_2 \in \mathcal{F}$ leads to two forbidden subgraphs for G (see graphs A and B in Fig. 3). □

The remaining part of this section is devoted to the proof of Theorem 3.1. We begin with the direction "(1) ⇒ (2)":

Proposition 3.1. *If a graph G is a 4-leaf power, then G is chordal and its critical clique graph $\mathrm{CC}(G)$ is $\mathcal{F}$-free.*

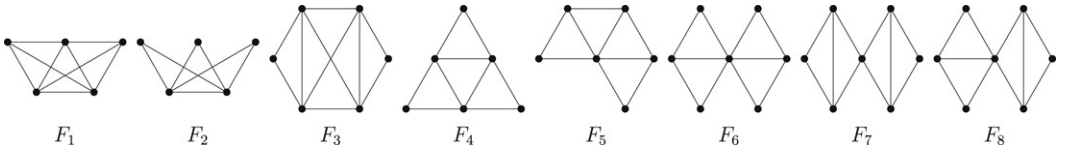

Fig. 2. The eight forbidden subgraphs for critical clique graphs of 4-leaf powers.

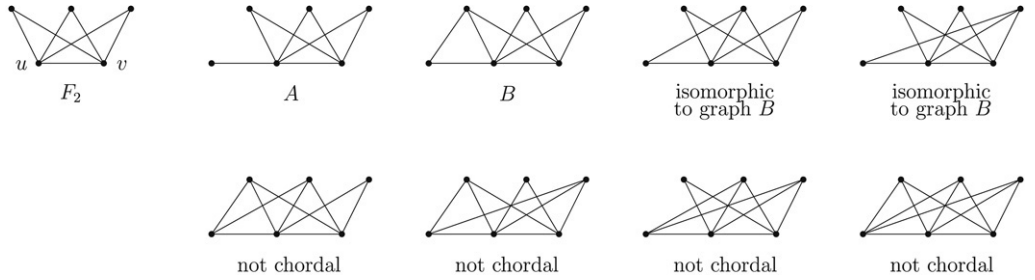

Fig. 3. The graph F_2 as an example for a forbidden subgraph in $\mathrm{CC}(G)$ and the corresponding forbidden subgraphs in G. Note that F_2 alone is not a forbidden subgraph in G, since it induces no F_2 in $\mathrm{CC}(G)$ when the two degree-4 vertices are in the same critical clique. There are sixteen forbidden subgraphs in G corresponding to F_2 (of which only two are chordal and pairwise nonisomorphic). The figure shows the eight possibilities in the case that there is a vertex that is adjacent to u but not to v.

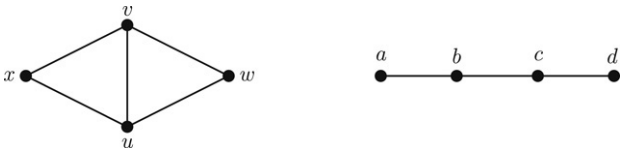

Fig. 4. The left graph is a subgraph of F_2 containing two nonadjacent nodes w and x. Nodes w and x have two common neighbors u and v. The only possible 2-Steiner root for this subgraph is shown in the right graph with $\{w, x\} = \{a, d\}$ and $\{u, v\} = \{b, c\}$.

Proof. If G is a leaf power, then G must be chordal [28]. With Lemma 2.1, it suffices to show that if $\mathrm{CC}(G)$ has a 2-Steiner root, then $\mathrm{CC}(G)$ is $\mathcal{F}$-free. In the following, we only show that a critical clique graph which has a 2-Steiner root contains no induced F_2. The proof for the other subgraphs in $\mathcal{F}$ is analogous.

Suppose that there is an induced F_2 in $\mathrm{CC}(G)$ and $\mathrm{CC}(G)$ has a 2-Steiner root T. Let u, v denote the two nodes having four neighbors in the induced F_2 and let w, x, y denote the other three nodes. Consider nodes w, x which are not adjacent in $\mathrm{CC}(G)$. Then, $d_T(w, x) \geq 3$. Since w and x have common neighbors u and v, we have that

$$\max\{d_T(u, w), d_T(u, x), d_T(v, w), d_T(v, x)\} \leq 2.$$

This implies that u and v lie on the path in T between w and x. Moreover, $d_T(w, x) < 4$ since, otherwise, there are no two distinct nodes on the path between w and x in T which have distance of at most two to both w and x in T. Therefore, $d_T(w, x) = 3$. See Fig. 4 for an illustration. With the same argument, for the nonadjacent nodes x, y, the path between them in T passes only u and v. Then, one of u and v has to be a common neighbor to w and y, and we have $d_T(w, y) = 2$. Since w and y are nonadjacent in F_2, this is a contradiction to T being a 2-Steiner root of $\mathrm{CC}(G)$. □

The reverse direction, i.e., "(2) ⇒ (1)", is technically more difficult.

Proposition 3.2. *If a graph G is chordal and its critical clique graph $\mathrm{CC}(G)$ is $\mathcal{F}$-free, then G is a 4-leaf power.*

Proof. If G is chordal, then $\mathrm{CC}(G)$ is also chordal. We show constructively that every $\mathcal{F}$-free and chordal critical clique graph indeed has a 2-Steiner root by using Algorithm SRG (Fig. 5). This algorithm reuses the constructions of [28]. For more details of the algorithm, see Section 3.1.

The correctness of the algorithm will be shown by the following three claims:

```
SRG(CC(G) = (V_C, E_C))
Input:  (F ∪ {C_4, C_5})-free critical clique graph CC(G) = (V_C, E_C)
Output: Pseudo Steiner root graph S of CC(G)
1  S ← ({b_c | c ∈ V_C}, ∅)
2  L ← list of all maximal cliques of CC(G)
3  while there is a maximal clique K in L which shares edges (c_1, c_2)
         and (c_1, c_3) with two other maximal cliques K' and K'' in CC(G):
4     Delete K from L
5     for c ∈ K, c ≠ c_1:
6        Insert an edge between b_{c_1} and b_c
7  while there is a maximal clique K in L which shares one edge (c_1, c_2)
         with one other maximal clique K' in CC(G):
8     Delete K from L
9     if K' is in L:
10       for c ∈ K, c ≠ c_1:
11          Insert an edge between b_{c_1} and b_c
12    else:
13       c' ← a node in K' \ K
14       if there is an edge (b_{c_1}, b_{c'}) in S:
15          for c ∈ K, c ≠ c_2:
16             Insert an edge between b_{c_2} and b_c
17       else:
18          for c ∈ K, c ≠ c_1:
19             Insert an edge between b_{c_1} and b_c
20 while there is a maximal clique K in L:
21    Delete K from L
22    Add a new node s_K into S
23    for c ∈ K:
24       Insert an edge between s_K and b_c
25 while there are at least two connected components S_1 and S_2 in S:
26    Add two new edge-connected Steiner nodes s_1 and s_2 to S and connect s_1
      by an edge to an arbitrary node in S_1 and s_2 to an arbitrary node in S_2
27 return S
```

Fig. 5. Algorithm to construct the pseudo Steiner root graph S of a critical clique graph $\mathrm{CC}(G)$. In the next section we show that there are at most $2 \cdot |E_C|$ maximal cliques in an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G) = (V_C, E_C)$.

(1) Every maximal clique K of an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G)$ is considered at least once by Algorithm SRG, and for every node pair u, v in K, a path of length at most two is generated between the corresponding nodes of u and v in the output graph.
(2) For an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G) = (V_C, E_C)$ Algorithm SRG outputs a graph with the following property: If two nodes $u, v \in V_C$ are not adjacent in $\mathrm{CC}(G)$, then the distance between the nodes corresponding to u and v in the output graph is at least three.
(3) For a chordal and $\mathcal{F}$-free critical clique graph $\mathrm{CC}(G)$ the output graph of Algorithm SRG is a tree.

The proofs of these claims are in Section 3.2. Together with Lemma 2.1, the claims prove Proposition 3.2. □

We say that a graph has the "distance property" if it fulfills Claims 1 and 2; note that exactly this distance property is required by Definition 2.3 for trees to be Steiner roots. The fixed-parameter algorithms for CLP4 in Section 4 also make use of these claims.

3.1. The algorithm SRG

We now present the algorithm used in the proof of Proposition 3.2. It extends a method by Lin et al. [28] for constructing 2-Steiner roots: While their algorithm only computes an output graph if the input graph has a 2-Steiner root and says "no" otherwise, our Algorithm SRG (Fig. 5) also generates an output graph with some guaranteed

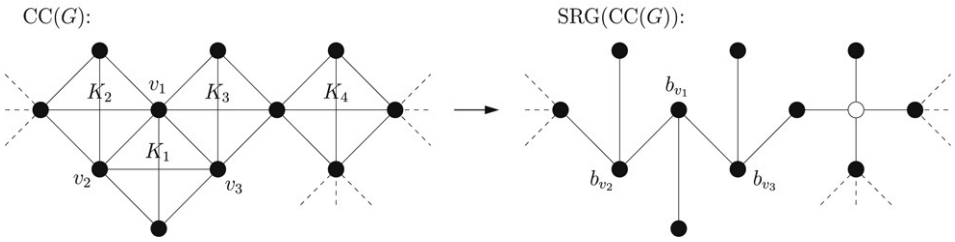

Fig. 6. Example of a subgraph of a critical clique graph CC(G) and the pseudo Steiner root graph computed for this subgraph. Algorithm SRG first considers the maximal clique K_1 with $c_1 = v_1$ (see Fig. 5) and inserts edges between b_{v_1} and the other nodes corresponding to K_1. Thereafter, the cliques K_2 and K_3 are considered. When considering K_4, Algorithm SRG inserts a Steiner node (drawn white).

properties for inputs that are $(\mathcal{F} \cup \{C_4, C_5\})$-free but nonchordal graphs. This will be of use for our fixed-parameter algorithms in Section 4.

For a given critical clique graph $\mathrm{CC}(G) = (V_C, E_C)$, Algorithm SRG constructs a *pseudo Steiner root graph* $S = (V', E')$ with $V' := A \cup B$, where $B := \{b_c \mid c \in V_C\}$ and $A \cap B = \emptyset$. The nodes in A and B are called *Steiner* and *non-Steiner* nodes, respectively. Each non-Steiner node one-to-one corresponds to a node in CC(G), whereas Steiner nodes do not correspond to nodes in CC(G). If CC(G) is $\mathcal{F}$-free and chordal, then S is a 2-Steiner root of CC(G). (The term "pseudo Steiner root graph" expresses that if the input graph is $(\mathcal{F} \cup \{C_4, C_5\})$-free but nonchordal, then the output S has some, but not all properties of a 2-Steiner root.)

The idea of the algorithm is to consider every maximal clique of the input graph CC(G) and to connect the corresponding nodes in the output graph to form a star. More specifically, if a maximal clique K in CC(G) has an edge e in common with another maximal clique K', then the node in the output graph corresponding to one of the endpoints of e is connected by edges with the other nodes corresponding to K and the node in the output graph corresponding to the other endpoint of e is connected by edges with the other nodes corresponding to K' (lines 3–19 in Fig. 5). If otherwise K has no edge in common with another maximal clique, a Steiner node s_K is inserted into the output graph, and every node corresponding to a node of K is connected by an edge with s_K (lines 20–24 in Fig. 5; see Fig. 6 for an example).

3.2. Correctness of the claims

Here, we show that the three claims used in the proof of Proposition 3.2 hold. To this end, we need the following four lemmas which show some specific properties of $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graphs. The proofs of the first two lemmas can be found in [28]. We call an edge shared by at least two maximal cliques a 2-*edge*.

Lemma 3.1. *In an $\{F_1\}$-free critical clique graph, two maximal cliques have at most two nodes in common.*

Lemma 3.2. *In an $\{F_1, F_2\}$-free critical clique graph, three maximal cliques have at most one node in common.*

Lemma 3.3. *In an $\{F_1, F_3, F_4, C_4\}$-free critical clique graph, if a maximal clique K contains two or more 2-edges, then there is exactly one node that is an endpoint of all 2-edges in K.*

Proof. If a maximal clique K of the critical clique graph CC(G) contains two or more 2-edges, then no two of them can be node-disjoint: Suppose that there are two 2-edges $e = (u_1, v_1)$ and $e' = (u_2, v_2)$ being node-disjoint. By Lemma 3.1, there exist two distinct maximal cliques K_1 and K_2 sharing with K edges e and e', respectively, and there are two nodes $u \in (K_1 \setminus K)$ and $v \in (K_2 \setminus K)$. Since CC(G) is $\{F_1\}$-free, none of the edges (u, u_2), (u, v_2), (v, u_1), and (v, v_1) is in E_C. Moreover, $(u, v) \notin E_C$; otherwise, there would be a C_4 induced by u, v, u_1, u_2 in CC(G). This implies that the edges e and e' together with u and v induce a forbidden subgraph F_3 which is a contradiction.

Consider a maximal clique K containing more than two 2-edges. Suppose that there are three distinct 2-edges such that there is no node which is an endpoint of all these three 2-edges. Any two of these three 2-edges have a common endpoint as shown above. Thus, these three 2-edges induce a triangle. With almost the same argument used above, we can show that this triangle together with three nodes which are from the three maximal cliques sharing these three edges with K, respectively, induce the forbidden subgraph F_4. This gives a contradiction. □

Lemma 3.4. *In an $\{F_1, F_2, F_5, F_6, C_4, C_5\}$-free critical clique graph $\mathrm{CC}(G)$, if two 2-edges share an endpoint v, then there is exactly one maximal clique K that contains the three endpoints of these two 2-edges. Moreover, K contains the endpoints of all 2-edges incident to v.*

Proof. The proof is by contradiction. Suppose that there are two 2-edges (v_i, v_j) and (v_j, v_k) in $\mathrm{CC}(G)$ that are not part of a common maximal clique, i.e., $(v_i, v_k) \notin E_C$. Let the two maximal cliques containing (v_i, v_j) be K_a and K_b, and let the two maximal cliques containing (v_j, v_k) be K_c and K_d. Since $(v_i, v_k) \notin E_C$, K_a, K_b, K_c, K_d are pairwise distinct. We consider the following two cases.

The first case is that there is a node v_b other than v_j that is contained in exactly one of K_a and K_b and in exactly one of K_c and K_d, say in K_b and K_d. Clearly, v_b is different from v_i and v_k. Let v_a be a node in $K_a \setminus K_b$ and v_c a node in $K_c \setminus K_d$. Node v_a cannot be identical to v_c; otherwise, v_a, v_k, v_b, v_i would induce a C_4. With the same argument, neither v_a and v_k nor v_c and v_i are adjacent. If v_a and v_c are adjacent, then v_a, v_c, v_k, v_b, v_i induce a C_5. If v_a and v_c are not adjacent, then $v_a, v_i, v_b, v_k, v_c, v_j$ induce an F_5.

The second case is that v_j is the only node that is contained in at least one of K_a and K_b and in at least one of K_c and K_d. Let v_a be a node in $K_a \setminus K_b$, v_b a node in $K_b \setminus K_a$, v_c a node in $K_c \setminus K_d$, and v_d a node in $K_d \setminus K_c$ with $(v_a, v_b) \notin E_C$ and $(v_c, v_d) \notin E_C$. Node v_i cannot be adjacent to v_c; otherwise, there would be a maximal clique containing v_i, v_j, v_c which shares edge (v_i, v_j) with the maximal cliques K_a and K_b, a contradiction to Lemma 3.2. With the same argument, none of (v_i, v_d), (v_k, v_a), and (v_k, v_b) can be in E_C. Since $\mathrm{CC}(G)$ is $\{F_5, C_4\}$-free, node v_a or node v_b is adjacent to neither v_c nor v_d. Then $v_i, v_a, v_b, v_j, v_c, v_d, v_k$ induce the forbidden subgraph F_6.

The uniqueness of the maximal clique that contains all the 2-edges follows from Lemma 3.1. □

Now, we are in the position to show the three claims.

Claim 1. *Every maximal clique K of an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G)$ is considered at least once by Algorithm SRG, and for every node pair u, v in K, a path of length at most two is generated between the corresponding nodes of u and v in the output graph.*

Proof. This claim follows directly from Lemmas 3.1–3.3 and the description of Algorithm SRG. □

Claim 2. *For an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph $\mathrm{CC}(G) = (V_C, E_C)$ Algorithm SRG outputs a graph with the following property: If two nodes $u, v \in V_C$ are not adjacent in $\mathrm{CC}(G)$, the distance between the nodes corresponding to u and v in the output graph is at least three.*

Proof. For two nonadjacent nodes v_i and v_k in $\mathrm{CC}(G)$, their corresponding non-Steiner nodes cannot be adjacent in the pseudo Steiner root graph S output by Algorithm SRG: An edge between two non-Steiner nodes can be inserted only by the first or the second while-loop of Algorithm SRG. However, the conditions of these two while-loops imply that v_i and v_k have to be adjacent in $\mathrm{CC}(G)$. In the following, we show by contradiction that the distance between the corresponding non-Steiner nodes of v_i and v_k cannot be two. Suppose that there is a path $b_{v_i}, b_{v_j}, b_{v_k}$ in S, where there is no edge between the non-Steiner nodes b_{v_i} and b_{v_k}. The node b_{v_j} cannot be a Steiner node since a Steiner node is only adjacent to non-Steiner nodes whose corresponding nodes in $\mathrm{CC}(G)$ induce a clique due to the third while-loop of Algorithm SRG. Hence, all three nodes are non-Steiner nodes.

By the description of Algorithm SRG, the edges (b_{v_i}, b_{v_j}) and (b_{v_j}, b_{v_k}) can be inserted only if there are two maximal cliques in $\mathrm{CC}(G)$ that contain the edges (v_i, v_j) and (v_j, v_k), respectively, and both maximal cliques contain at least one 2-edge. We distinguish three cases based on whether the 2-edges in these two maximal cliques have v_j as an endpoint or not. The three cases are illustrated in Fig. 7. In the first case where v_j is not the endpoint of the two 2-edges (v_a, v_c) and (v_d, v_f), the maximal clique containing v_i and v_j is K_2 with $v_i \in \{v_a, v_c\}$ and the maximal clique containing v_j and v_k is K_3 with $v_k \in \{v_d, v_f\}$. In the second case where v_j is the endpoint of one 2-edge, we can without loss of generality assume that the 2-edge in the maximal clique containing v_j and v_k has v_j as one endpoint. Then, v_k can be one of v_d, v_e, and v_f. In the third case, both of the two 2-edges have v_j as one endpoint. Here, $v_i \in \{v_a, v_b, v_c\}$ and $v_k \in \{v_d, v_e, v_f\}$.

In each of these three cases, one can make a further case distinction based on whether two nodes in Fig. 7 which are not v_i, v_j, or v_k can be identical or whether there are additional edges between the nodes. However, in each of these cases, either we can find one of the induced subgraphs in $\mathcal{F} \cup \{C_4, C_5\}$ in $\mathrm{CC}(G)$, or we can conclude that the algorithm cannot create both edges (b_{v_i}, b_{v_j}) and (b_{v_j}, b_{v_k}) in S. This completes the proof. □

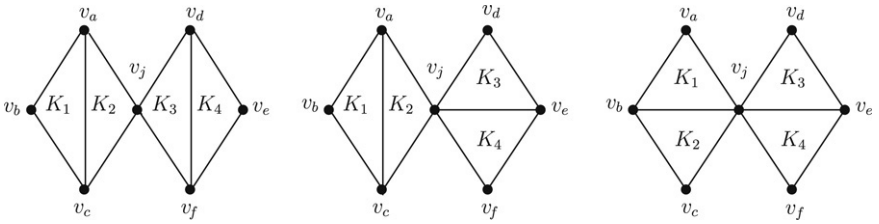

Fig. 7. Illustration of the three cases in the proof of Claim 2.

In order to show Claim 3, we need the following lemma. The fixed-parameter algorithms for CLP4 in Section 4 also make use of this lemma.

Lemma 3.5. *For an* $(\mathcal{F} \cup \{C_4, C_5\})$*-free critical clique graph* $\mathrm{CC}(G)$ *the output graph of Algorithm SRG contains no cycle of length less than seven.*

Proof. *Triangles.* We start with showing that there is no triangle in the pseudo Steiner root graph S constructed by Algorithm SRG. Hence, for the purpose of contradiction, suppose that S contains a triangle. Since the algorithm does not insert an edge between two Steiner nodes, at most one of the three nodes of this triangle can be a Steiner node. We distinguish two cases:

Case 1: One node of the triangle is a Steiner node. Then let s, b_{v_1}, and b_{v_2} be the nodes of the triangle where s is the Steiner node. On the one hand, by the construction of the third while-loop of Algorithm SRG, the neighbors of s in S (in particular b_{v_1} and b_{v_2}) correspond to the nodes of a maximal clique K in $\mathrm{CC}(G)$ that shares no edge with another maximal clique. On the other hand, the edge between b_{v_1} and b_{v_2} can only have been inserted in the first or second while-loop of Algorithm SRG. Hence, there must exist a maximal clique K' in $\mathrm{CC}(G)$ that contains both v_1 and v_2 and that contains a 2-edge with endpoint v_1 or v_2. Then, K and K' share the edge (v_1, v_2) and we have a contradiction.

Case 2: The triangle contains no Steiner node. Then let b_{v_1}, b_{v_2}, and b_{v_3} be the nodes of the triangle. For the edge (b_{v_1}, b_{v_2}) to be inserted, there must be a maximal clique K_1 in $\mathrm{CC}(G)$ with $v_1, v_2 \in K_1$ for which the algorithm selects one of the nodes b_{v_1} or b_{v_2}, say b_{v_1}, to be connected with all other nodes corresponding to the nodes of K_1.

Analogously, the edge (b_{v_2}, b_{v_3}) can only be inserted if there exists a maximal clique $K_2 \neq K_1$ in $\mathrm{CC}(G)$ with $v_2, v_3 \in K_2$ for which the algorithm selects one of the nodes b_{v_2} or b_{v_3} and connects it with all other nodes corresponding to the nodes of K_2.

In S the three nodes b_{v_1}, b_{v_2}, and b_{v_3} all have a distance of at most two to every node of K_1 as well as to every node of K_2. But then, by Claims 1 and 2, the nodes v_1, v_2, and v_3 all are part of both K_1 and K_2—a contradiction to Lemma 3.1.

Cycles of length 4. Next, we show that there is no cycle of length 4 in S. Suppose, for the sake of a contradiction, that S contains a length-4 cycle, but no triangle. Here we have to distinguish three cases:

Case 1: Two nodes of the cycle are Steiner nodes. Then let s_1, b_{v_1}, s_2, and b_{v_2} be the nodes of the cycle, where s_1 and s_2 are the Steiner nodes. By the construction of the third while-loop of Algorithm SRG, the neighbors of a Steiner node correspond to the nodes of a maximal clique K in $\mathrm{CC}(G)$ that shares no edge with another maximal clique. But then v_1 and v_2 both belong to two such maximal cliques, which is a contradiction.

Case 2: One node of the cycle is a Steiner node. Then let s, b_{v_1}, b_{v_2}, and b_{v_3} be the nodes of the cycle, where s is the Steiner node. On the one hand, the neighbors of s in S (in particular b_{v_1} and b_{v_3}) correspond to the nodes of a maximal clique K in $\mathrm{CC}(G)$ that shares no edge with another maximal clique and that does not contain v_3. On the other hand, the distance between each two of the nodes b_{v_1}, b_{v_2}, and b_{v_3} in S is at most two, which implies, due to Claim 2, that there is a maximal clique in $\mathrm{CC}(G)$ that contains v_1, v_2 and v_3—a contradiction to the fact that K shares no edge with another maximal clique.

Case 3: The cycle contains no Steiner node. Because the distance between each two of the four nodes of the cycle is at most two, there is a clique K in $\mathrm{CC}(G)$ that contains all these four nodes due to Claim 2. But then, since the Algorithm SRG must consider K at least once, there must be a node (which can be a Steiner node) in S that is adjacent to all four nodes of the cycle, which implies that S contains a triangle.

Cycles of length 5. In order to show that there is no cycle of length 5 in S, suppose, for the sake of a contradiction, that S contains a length-5 cycle, but no cycle of length less than five. Since the algorithm does not insert an edge between two Steiner nodes, at most two of the five nodes of this cycle can be Steiner nodes, and these Steiner nodes cannot be adjacent. Hence, let x_1, b_{v_1}, x_2, b_{v_2}, and b_{v_3} be the nodes of the cycle where each of x_1 and x_2 can be a Steiner node. Because the distance between each two of the nodes b_{v_1}, b_{v_2}, and b_{v_3} is at most two, we can find a triangle in S with the same argumentation as in the last case for cycles of length 4.

Cycles of length 6. For the sake of a contradiction, suppose that S contains a length-6 cycle, but no cycle of length less than six. We distinguish two cases:

Case 1: The cycle consists of the nodes s_1, b_{v_1}, b_{v_2}, s_2, b_{v_3}, and b_{v_4}, where s_1 and s_2 are Steiner nodes. Due to Claims 1 and 2, the nodes v_1, v_2, v_3, and v_4 induce a C_4 in $\mathrm{CC}(G)$—a contradiction.

Case 2: The cycle consists of the nodes x_1, b_{v_1}, x_2, b_{v_2}, x_3, and b_{v_3}, where each of x_1, x_2, and x_3 can be a Steiner node. Because the distance between each two of the nodes b_{v_1}, b_{v_2}, and b_{v_3} is at most two, we can find a triangle or an induced C_4 in S with an argumentation similar to the last case for cycles of length 4. □

Although for $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graphs $\mathrm{CC}(G)$ the output graph of Algorithm SRG contains no cycle of length at most 6, it can still contain cycles of greater length; consider, for example, the output graph when the input $\mathrm{CC}(G)$ is a C_6: in this case, the algorithm outputs a C_{12}. However, if $\mathrm{CC}(G)$ is not only $(\mathcal{F} \cup \{C_4, C_5\})$-free but chordal, Claim 3 states that the output of Algorithm SRG contains no cycle at all. We will prove this claim now.

Claim 3. *For a chordal and $\mathcal{F}$-free critical clique graph $\mathrm{CC}(G)$ the output graph of Algorithm SRG is a tree.*

Proof. Suppose that the output graph S on input $\mathrm{CC}(G)$ is not a tree, and $\mathrm{CC}(G)$ is $(\mathcal{F} \cup \{C_4, C_5\})$-free. Consider the shortest cycle Q of S. Let $\mathrm{CC}(G)_Q$ denote the subgraph of $\mathrm{CC}(G)$ which is induced by the nodes of $\mathrm{CC}(G)$ whose corresponding nodes are on Q. With Lemma 3.5, Q has a length of at least 7. Since no two Steiner nodes are adjacent, $\mathrm{CC}(G)_Q$ has at least four nodes. Furthermore, $\mathrm{CC}(G)_Q$ has a Hamiltonian cycle passing through all its nodes due to Claim 1. We "embed" $\mathrm{CC}(G)_Q$ into Q by identifying the nodes of $\mathrm{CC}(G)_Q$ with their corresponding non-Steiner nodes in Q. There can be some new edges in this embedding which are not in Q and which are between two non-Steiner nodes with a distance of two on Q due to Claims 1 and 2. However, since each triangulation of a cycle with length of at least seven has to contain a chord between two nodes with a distance of at least three on this cycle, this embedding is not a chordal graph. Moreover, it is easy to observe that every hole in this embedding solely consists of non-Steiner nodes. This implies that $\mathrm{CC}(G)_Q$ is not chordal. As a consequence, $\mathrm{CC}(G)$ is then not chordal. □

4. Fixed-parameter tractability of CLP4

In this section, we show the fixed-parameter tractability of CLP4 Edge Deletion, CLP4 Edge Insertion, and CLP4 with respect to the parameter "number of edge editing operations" r. The basic approach resembles our previous work for CLP3 [13]; however, for the case of CLP4 Edge Deletion new, more intricate methods are necessary. Therefore, we focus on the CLP4 Edge Deletion case in this section.[3]

Note that graphs that have 3-leaf roots have a characterization similar to that of Theorem 3.1: they are graphs that are chordal and contain none of the induced subgraphs "bull", "dart", and "gem" [13]. Therefore, the basic idea for CLP3 Edge Deletion as well as for CLP4 Edge Deletion is to use the forbidden subgraph characterization in a depth-bounded search tree algorithm: find a forbidden subgraph, and recursively branch into several cases according to the possible edge deletions that destroy the forbidden subgraph. If we can upper-bound the number of branching cases by a function depending only on r, since the depth can be bounded from above by r, we obtain a run time that proves fixed-parameter tractability.

Since the forbidden subgraph characterization from Theorem 3.1 for the critical clique graph $\mathrm{CC}(G)$ is much simpler than the implied characterization for G (Corollary 3.1), we would like to apply modifications directly on $\mathrm{CC}(G)$. This is possible by the following lemma, which is a straightforward extension of Lemma 4 in [13].

Lemma 4.1. *For a graph G, there is always an optimal solution for CLP4 that is represented by edge editing operations on $\mathrm{CC}(G)$. That is, one can find an optimal solution that does not delete any edges within a critical clique; furthermore, in this optimal solution, between two critical cliques either all or no edges are inserted or deleted.*

[3] Note that for the edge insertion variant the fixed-parameter tractability immediately follows from Theorem 3.1 and results of Cai [7].

Now, working with $CC(G) = (V_C, E_C)$ instead of G has two consequences: First, a deletion of an edge e in $CC(G)$ can represent several edge deletions in G. Consider an edge e in $CC(G)$ between two nodes that represent critical cliques of sizes c_1 and c_2. Deleting e implies deleting all $c_1 \cdot c_2$ edges between the vertices of the critical cliques in G. Therefore, we give the edge e the weight $c_1 \cdot c_2$. Note that this means that an edge modification on $CC(G)$ can decrease the parameter r in the depth-bounded search tree algorithm by more than one. Second, if two adjacent nodes in $CC(G)$ obtain an identical neighborhood after deleting edges in $CC(G)$, then $CC(G)$ needs to be updated, since each node in $CC(G)$ has to represent a critical clique in G. In this situation a *merge* operation is needed, which replaces these nodes in $CC(G)$ by a new node with the same neighborhood as the original nodes. Note that a hole can be destroyed by merge operations if we add or delete edges to make its vertices have the same neighborhood. In the following, we assume that after each modification of $CC(G)$, all pairs of nodes in $CC(G)$ are checked as to whether a merge operation between them is required. This can be done in linear time by modifying G accordingly and computing the critical clique graph $CC(G)$ of the modified graph G [28].

The main obstacle in obtaining fixed-parameter tractability for both CLP3 Edge Deletion and CLP4 Edge Deletion is that the holes in $CC(G)$ can have arbitrary length, and, therefore, one cannot simply find some hole and branch for each edge of the hole that is to be deleted—the size of the search tree would not be a function depending on r. For CLP3 Edge Deletion, the key observation is that the critical clique graph $CC(G)$ of a graph G containing neither a bull nor a dart nor a gem nor a C_4 contains no triangles. This allows to show that, after destroying the forbidden subgraphs bull, dart, gem, and C_4 in G, no hole in $CC(G)$ can be "accidentally" destroyed by merge operations between its nodes and, therefore, one has to delete at least one edge of every hole. Since moreover making a triangle-free graph chordal means to make it a forest, a minimum weight set of edges to be deleted to make $CC(G)$ chordal can be obtained in polynomial time by searching for a maximum weight spanning tree. Unfortunately, there *can* be triangles in an $\mathcal{F}$-free (Fig. 2) $CC(G)$ as we obtain it for CLP4 after deleting the forbidden subgraphs. Thus, the main technical contribution of this section is to show how to circumvent these difficulties by new, more sophisticated techniques than that required for CLP3 Edge Deletion.

The idea is to examine the output graph $SRG(CC(G))$ of Algorithm SRG (Fig. 5) for the critical clique graph $CC(G)$. If it is a tree, we are done. Otherwise, the output is a pseudo Steiner root graph S that contains a cycle which corresponds to a hole in $CC(G)$. By repeatedly deleting degree-1 nodes and contracting the middle node of three consecutive degree-2 nodes in S we get a graph S' in which there is no path that consists of three or more consecutive degree-2 nodes. By finding the shortest cycle in this reduced graph S', whose length is bounded by $O(\log |V|)$ due to a result of Erdős and Pósa [16], we can obtain an "FPT hole" in $CC(G)$, that is, a hole for which we can bound the number of possibilities to delete edges to get rid of the hole in an optimal way by $O(\log |V|)$ (see Fig. 8). This suffices to upper-bound the search tree size by a function only depending on r.

For the pseudocode of this algorithm, which is presented in Fig. 9, we introduce some notation for the mapping between the nodes of a critical clique graph and the nodes of its pseudo Steiner root graph.

Definition 4.1. Consider a critical clique graph $CC(G) = (V_C, E_C)$ and a pseudo Steiner root graph $S = (V_S, E_S)$ constructed by Algorithm SRG for $CC(G)$. For $v \in V_C$ we use $S(v)$ to denote the node from V_S that corresponds to v, and for $v_S \in V_S$, we define $S^{-1}(v_S)$ as the node in V_C corresponding to v_S if v_S is a non-Steiner node, or $\perp$ if v_S is a Steiner node. We extend this notation to sets: for $V'_C \subseteq V_C$, $S(V'_C) := \{S(v) \mid v \in V'_C\}$, and for $V'_S \subseteq V_S$, $S^{-1}(V'_S) := \{S^{-1}(v) \mid v \in V'_S\}$.

4.1. *Correctness of the* CLP4 Edge Deletion *algorithm*

To define the branching set D in line 18 of Algorithm CLP4Del-Branch, we need some notation.

Definition 4.2. A *big node* is a node of a pseudo Steiner root graph S that is not deleted by the data reduction in lines 11–19 of Algorithm CLP4Del-Branch (Fig. 9) and that has degree at least 3 in the constructed pseudo Steiner root graph S' (see Fig. 10).

For a cycle Q in a pseudo Steiner root graph S as constructed by Algorithm CLP4Del-Branch in line 16, let $v_0, \ldots, v_{q-1}$ be the big nodes in Q, ordered by their appearance in Q, and for every node v_i with $0 \leq i < q$ let P_i be the path in Q between v_i and $v_{(i+1) \bmod q}$.

With P_i^+ we denote the path P_i plus its attached trees, that is, the maximal set of nodes in S such that P_i^+ contains the nodes of P_i and such that P_i^+ induces a connected component in $S \setminus \{v_i, v_{(i+1) \bmod q}\}$.

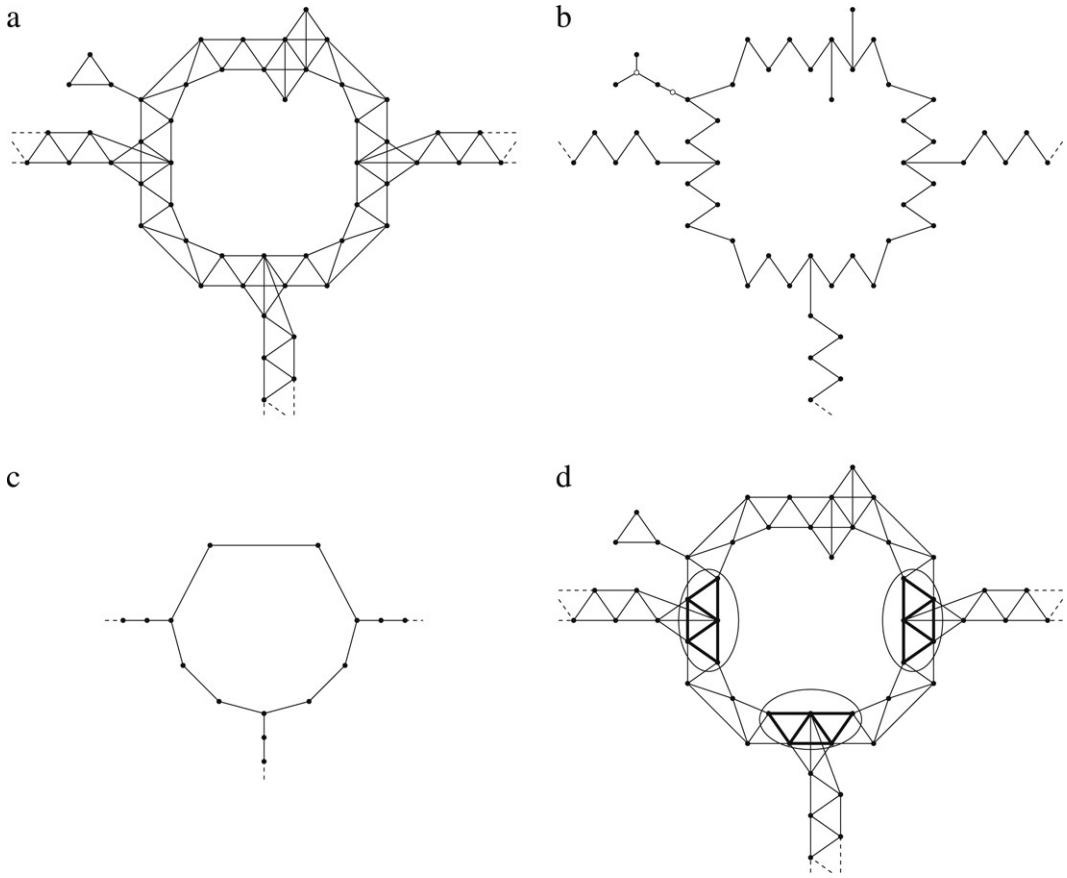

Fig. 8. Illustration of finding and destroying holes in an $(\mathcal{F} \cup \{C_4, C_5\})$-free critical clique graph: (a) A nonchordal critical clique graph $\mathrm{CC}(G)$. (b) The pseudo Steiner root graph S constructed by Algorithm SRG for $\mathrm{CC}(G)$. (c) The reduced pseudo Steiner root graph S' constructed in lines 10–14 of Fig. 9. (d) The sets marked with an ellipsis correspond to the degree-3 nodes in S'. Our algorithm for CLP4 EDGE DELETION either deletes one of the bold edges or it deletes a minimum weight set of edges between two of the node sets marked with an ellipsis (Lemma 4.3).

We further denote with A_i the *big node areas* that are defined as

$$A_i := S^{-1}(\{v \in Q \mid d_S(v_i, v) \leq 2\}) \setminus \{\perp\}.$$

The following lemma will help us to show that the cycle Q determined by Algorithm CLP4DEL-BRANCH (Fig. 9) in line 16 indeed induces at least one hole in $\mathrm{CC}(G)$.

Lemma 4.2. *Consider a cycle Q in a pseudo Steiner root graph S as constructed by Algorithm* CLP4DEL-BRANCH *(Fig. 9) in line 16. Let $v_0, \ldots, v_{p-1}$ be the nodes of Q, ordered by their appearance in Q. Then there is no edge $(S^{-1}(v_i), S^{-1}(v_j))$ with $0 \leq i, j < p$ in $\mathrm{CC}(G)$ such that v_i and v_j have a distance of more than 2 on Q.*

Proof. Assume there is such an edge $(S^{-1}(v_i), S^{-1}(v_j))$ in $\mathrm{CC}(G)$. Without loss of generality, let $j > i$. Because of the "distance property" of S (more specifically, because of Claim 2), the distance in S between v_i and v_j is at most two, and, hence, there have to exist at least three paths from v_i to v_j in S: a path $P_1 = v_i, v_{i+1}, \ldots, v_{j-1}, v_j$ consisting of at least four nodes, a path $P_2 = v_i, v_{(i-1) \bmod p}, \ldots, v_{(j+1) \bmod p}, v_j$ consisting of at least four nodes, and a third path P_3 consisting of at most three nodes (including v_i and v_j). Note that P_1 and P_2 together form the cycle Q. Since there are three paths from v_i to v_j, one can easily see that v_i and v_j must be big nodes. Without loss of generality, let the node in P_3 between v_i and v_j, if such a node exists, not be a part of P_2.

The paths in S' corresponding to P_1 and P_2 consist of at least four nodes (including v_i and v_j), because the data reduction which transforms S to S' never transforms a path that connects two big nodes and that consists of at least four nodes into a path that consists of less than four nodes. The path in S' corresponding to P_3, however, has length

```
    CLP4DEL-BRANCH(G, r)
    Input:  A graph G = (V, E) and an integer r
    Output: A set of at most r edges in G whose removal makes G a 4-leaf power,
            or nil if no such set exists
1   if r < 0: return nil
2   Compute CC(G)
3   if CC(G) contains an induced forbidden subgraph F ∈ F ∪ {C4, C5}:
4     for each edge e in F:
5       X ← CLP4DEL-BRANCH(CC-DEL(G, {e}), r−CC-WEIGHT(G, {e}))
6       if X ≠ nil: return X ∪ {e}
7     return nil
8   S ← SRG(CC(G))
9   if S is a tree: return ∅
10  S' ← S
11  while there is a degree-1-node u in S':
12    delete u
13  while there is a path (u, v, w) of three degree-2-nodes in S':
14    delete v and insert an edge between u and w
15  Q' ← shortest cycle in S'
16  Q ← cycle in S corresponding to Q'
17  H ← S^{-1}(Q) \ {⊥}
18  Determine a set D (see Lemma 4.3) of edge sets in CC(G)[H] such that at
    least one d ∈ D is a subset of an optimal solution
19  for d ∈ D:
20    X ← CLP4DEL-BRANCH(CC-DEL(G, d), r−CC-WEIGHT(G, d))
21    if X ≠ nil: return X ∪ d
22  return nil
```

Fig. 9. Algorithm for CLP4 EDGE DELETION. The lines 3–7 recursively try all possibilities to destroy forbidden subgraphs, if existing (line 8 is only reached if in the current recursive call of the algorithm no forbidden subgraph was found in $CC(G)$). The lines 10–14 construct the graph S'. The lines 15–16 search for the "FPT hole", and the lines 17–22 try all possibilities that lead to the destruction of this hole. The subroutine CC-DEL(G, d) takes a graph G and a set d of edges in CC(G) as input. For every edge $(K_1, K_2) \in d$, all edges from G that have one endpoint in the clique represented by K_1 and the other endpoint in the clique represented by K_2 are deleted by CC-DEL(G, d). The function CC-WEIGHT(G, d) returns the sum of the weights of the edges in d.

at most three. Therefore, the cycle Q' which is the cycle in S' corresponding to Q (Fig. 9) cannot be a shortest cycle in S', because the cycle in S consisting of the paths P_1 and P_3 corresponds to a cycle in S' that is shorter than Q'. This is a contradiction to the claim that Algorithm CLP4DEL-BRANCH always chooses a shortest cycle in S' in line 15. □

The main observation that helps to bound the number of branching cases and, hence, leads to our fixed-parameter algorithm is that for a cycle Q in a pseudo Steiner root graph S the number of branching cases is independent of the lengths of the paths in Q between the big nodes: If we want to disconnect two big node areas, then it is always optimal to take an edge set with minimum weight whose removal disconnects the two big node areas. Such an edge set can be found in polynomial time by maximum flow techniques.

Using the notion of big node areas, we can conveniently give a precise characterization of the branching set.

Lemma 4.3. *Assume that in line* 18 *of* CLP4DEL-BRANCH *the branching set D is chosen as follows: Either delete an edge in a big node area, that is, an edge (u, v) with $u, v \in A_i$ for some $0 \leq i < q$, or delete a set of edges*

$$\text{MINCUT}\left(\text{CC}(G)[S^{-1}(P_i^+) \setminus \{\bot\}],\, A_i,\, A_{(i+1) \bmod q}\right),$$

that is, delete a minimum weight set of edges such that all paths between two neighboring big node areas are destroyed. Then the branching set D contains at least one subset of an optimal solution.

Proof. Consider an arbitrary optimal solution X for CLP4 EDGE DELETION. If X contains an edge from a big node area, then we are done. Consider therefore the case that X does not contain an edge from a big node area. We show

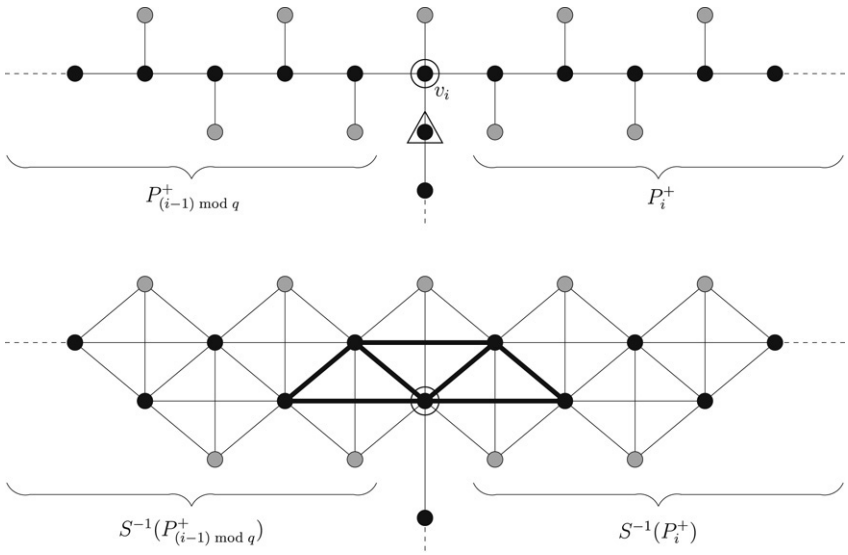

Fig. 10. Illustration for Definition 4.2. The upper picture shows a part of pseudo Steiner root graph S. The encircled node v_i represents a big node; black nodes represent nodes that are part of a cycle in S. The grey nodes are deleted by the data reduction in lines 11–14 of Algorithm CLP4DEL-BRANCH. The only Steiner node in this example is the node marked with a triangle. The lower picture shows the corresponding part of $\mathrm{CC}(G)$. The edges drawn with bold lines are those between vertices of the big node area A_i.

that there is some i such that X disconnects A_i and $A_{(i+1) \bmod q}$. It is then easy to see that an optimal solution that disconnects A_i and $A_{(i+1) \bmod q}$ does so by a minimum edge cut, concluding the proof.

Assume therefore now that X does not contain an edge in a big node area and that it does not disconnect any A_i and $A_{(i+1) \bmod q}$. After deleting the edges in X from $\mathrm{CC}(G)$, we show that $\mathrm{CC}(G)$ still contains a hole. Observe that if $A_i \cap A_{(i+1) \bmod q} \neq \emptyset$ for some $0 \leq i < q$, then X cannot disconnect A_i and $A_{(i+1) \bmod q}$. Thus, we only examine the case that for every $i \neq j$ the big node areas A_i and A_j have no node in common.

First we show that there is a hole Z in the unmodified $\mathrm{CC}(G)$ that contains only nodes from $H := S^{-1}(Q) \setminus \{\perp\}$. Assume for the sake of contradiction that there is no such hole, that is, $\mathrm{CC}(G)[H]$ is chordal. Construct a graph R from Q by inserting edges between each two non-Steiner nodes whose corresponding nodes in $\mathrm{CC}(G)[H]$ are adjacent. The resulting graph R has to be chordal since, compared to the chordal graph $\mathrm{CC}(G)[H]$, every Steiner node in R is adjacent to only two non-Steiner nodes which are directly connected by an edge. By Lemma 4.2, all edges added into R are the chords of Q between two non-Steiner nodes with a distance of two in Q. However, each triangulation of a cycle with length of at least seven has to contain a chord between two nodes with a distance of at least three on this cycle. Since by Lemma 3.5 Q has a length of at least seven, we have a contradiction and conclude that the hole Z does in fact exist. We now go on to show that a "related" hole still exists in the modified $\mathrm{CC}(G)$.

By our assumption of nonoverlapping big node areas, the hole Z contains at least one node from every big node area A_i with $0 \leq i < q$, since otherwise we obtain a contradiction to Lemma 4.2. Now we can easily show that, after deleting the edges of X from $\mathrm{CC}(G)$, the resulting critical clique graph still contains a hole: If an edge in Z between two big node areas A_i, $A_{(i+1) \bmod q}$ is deleted by X, we can replace $Z \cap P_i$ by a path connecting A_i and $A_{(i+1) \bmod q}$ whose nodes are all from P_i^+. Clearly, Z is still a cycle after this modification.

Next we show that Z does not contain any new chords. For this, consider a node u from such a replacement path, that is, $S(u) \in P_i^+ \setminus P_i$ for some $0 \leq i < q$, and let v be the node in P_i that has the smallest distance to u. We show that u can only be connected to a node in P_i^+ or to a node w where $S(w)$ is a neighbor of $S(v)$ in S. Assume for the sake of contradiction that there is an edge (u, w) in $\mathrm{CC}(G)$ such that $S(w)$ is not a neighbor of $S(v)$. Then $S(u)$ and $S(w)$ have a distance of at most two in S due to the "distance property" (Claims 1 and 2) of S. Hence, there exists a path from $S(u)$ to $S(w)$ in S that does not contain $S(v)$. But then, by the definition of P_i^+, the node $S(w)$ would be in P_i^+.

Now, a similar argumentation to that used to show the existence of the hole Z shows that $S^{-1}(\{v_0\} \cup P_0^+ \cup \cdots \cup \{v_{q-1}\} \cup P_{q-1}^+) \setminus \{\perp\}$ induces a hole in $\mathrm{CC}(G)$. □

We summarize the findings of this section in the following proposition.

Proposition 4.1. *Algorithm* CLP4DEL-BRANCH *(Fig. 9) correctly solves* CLP4 EDGE DELETION.

Proof. The algorithm is organized as a standard search tree algorithm. By Theorem 3.1, there is at least one branch that will lead to an optimal solution when branching in line 5, and by Lemma 4.3, there is at least one branch that will lead to an optimal solution when branching in line 20. □

It remains to show the complexity of CLP4DEL-BRANCH.

4.2. Complexity of the CLP4 EDGE DELETION algorithm

All steps within a single invocation of CLP4DEL-BRANCH can be done in polynomial time. We therefore focus on the number of recursive calls. In line 4, there can be at most 10 recursive calls corresponding to at most 10 edges to delete in a forbidden subgraph (for example F_3 in Fig. 2); as we will see, this is dominated by the number of recursive calls in line 20 for destroying a long cycle.

A well-known result by Erdős and Pósa [16] states that any graph with minimum vertex degree at least 3 has a cycle of length at most $2\log n + 1$, where n denotes the number of graph vertices. Using this result we can give an upper bound on the size of the shortest cycle in S' and show the following lemma:

Lemma 4.4. *When choosing* D *in line* 18 *of Algorithm* CLP4DEL-BRANCH *as described by Lemma 4.3, we can upper-bound its size by* $|D| \leq 96 \cdot \log |V| + 24$.

Proof. Consider a cycle Q in a pseudo Steiner root graph S as constructed by Algorithm CLP4DEL-BRANCH (Fig. 9) in line 16.

First, we show an upper bound for the number of big nodes in Q. It follows directly from the result of Erdős and Pósa [16] that there is a cycle in S' which contains at most $2\log |V_S| + 1$ nodes of degree at least three, because the graph resulting from contracting all degree-2 nodes in S' contains only nodes of degree at least three, and re-inserting the degree-2-nodes into the shortest cycle of this graph yields the claimed cycle in S'. Moreover, the data reduction which transforms S to S' guarantees that there can be at most two degree-2-nodes between each pair of nodes of degree at least three in S'. Hence, the shortest cycle Q' in S' has a length of at most $6\log |V_S| + 3$, bounding thereby the number of big nodes in Q by the same number. Note that $6\log |V_S| + 3 \leq 6\log(|V_C| + |E_C|) + 3 \leq 12\log |V_C| + 3$ since the number of Steiner nodes in S is bounded by the number of maximal cliques in $\mathrm{CC}(G)$ that do not share edges with other maximal cliques.

Next, for every big node v_i in Q we count the edges (u, v) with $u, v \in A_i$. From the "distance property" (Claims 1 and 2) of the pseudo Steiner root graph S we know that the only edges in $\mathrm{CC}(G)$ are those between nodes whose corresponding nodes in S are at distance one or two. Then there can be at most seven edges in $\mathrm{CC}(G)$ with both endpoints in A_i, because A_i consists of at most five nodes whose corresponding nodes in S form a path.

Altogether, the set D contains, for each of the at most $12\log |V| + 3$ big nodes in Q, seven edges between nodes of A_i plus one minimum weight edge set disconnecting the nodes in A_i from those in $A_{(i+1) \bmod q}$. This leads to the bound $|D| \leq 96 \cdot \log |V| + 24$. □

4.3. Fixed-parameter tractability results

Using Lemma 4.4, we arrive at the following central result.

Theorem 4.1. CLP4 EDGE DELETION *with* r *edge deletions allowed is fixed-parameter tractable with respect to* r.

Proof. By Proposition 4.1, Algorithm CLP4DEL-BRANCH correctly solves CLP4 EDGE DELETION. By Lemma 4.4 and the fact that the height of the search tree is bounded from above by r, it runs in $(96 \cdot \log |V| + 24)^r \cdot |V|^{O(1)} = c^r \cdot (r \log r)^r \cdot n^{O(1)}$ time for a constant c (the equality holds because $(\log n)^r \leq (3r \log r)^r + n$ for all values of n and r). □

With Theorem 4.1 and using the same techniques as applied for CLP3 EDGE INSERTION and CLP3 [13], we achieve the following result:

Theorem 4.2. 1. CLP4 EDGE INSERTION *with* r *edge insertions allowed is fixed-parameter tractable with respect to* r.

2. CLP4 *with* r *edge insertions and deletions is fixed-parameter tractable with respect to* r.

For CLP4 EDGE INSERTION, this follows from the fact that when we encounter a "long" induced cycle, that is, a cycle whose length is not bounded by a function depending only on r, we know the instance cannot be solved, because in order to destroy a cycle of length q with edge insertions, we have to insert $\Omega(q)$ edges (see [13]). For CLP4, we first get rid of the "short" cycles by branching; after that, inserting edges is useless, and we can continue as for CLP4 EDGE DELETION. In order to consider all "short" cycles before the "long" cycles, we modify Algorithm CLP4DEL-BRANCH as follows: In Line 10, to every edge of S' a weight equal to 1 is assigned; in Line 14, the newly inserted edge (u, w) gets a weight that is equal to the sum of the weights of the edges (u, v) and (v, w); and in Line 15, we search for a cycle with minimum weight in S' instead for a shortest cycle.

5. Concluding remarks

Our fixed-parameter algorithm constitutes the first positive algorithmic result for CLOSEST 4-LEAF POWER. To the best of our knowledge, so far results in this direction are only obtained for the simpler problems CLOSEST 2-LEAF POWER [2,10,19] and CLOSEST 3-LEAF POWER [13]. Besides improving on our running times—so far our algorithms are probably of purely theoretical interest—it would be challenging to study the fixed-parameter tractability of CLOSEST 5-LEAF POWER. Note that there is a recent linear-time algorithm for 5-LEAF POWER [8]. As long as it remains open to determine the complexity of k-LEAF POWER for $k \geq 6$, it seems to make little sense to study the more general CLOSEST k-LEAF POWER for this case. Given our new results, it is of particular interest to attack the open problem of finding good polynomial-time approximation algorithms for CLOSEST 3-LEAF POWER and CLOSEST 4-LEAF POWER. The only known result in this direction is a factor-2.5 approximation algorithm for CLOSEST 2-LEAF POWER [1,2,10,37], the by far simplest of these problems. Moreover, problem kernelization results for CLOSEST 3-LEAF POWER and CLOSEST 4-LEAF POWER as such for CLOSEST 2-LEAF POWER [17,20] would be highly desirable.

Also the CLOSEST k-TREE POWER problems as introduced by Kearney and Corneil [22] deserve further investigations. Note that they only state a straightforward solution that calls the (exact) tree power recognition algorithm $O(n^r)$ times, thus exhaustively trying all possibilities. This clearly does not lead to fixed-parameter tractability since the parameter r (number of edge modifications) appears in the exponent of the polynomial.

References

[1] N. Ailon, M. Charikar, A. Newman, Proofs of conjectures in Aggregating inconsistent information: Ranking and clustering. Technical Report TR-719-05, Department of Computer Science, Princeton University, 2005.
[2] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Mach. Learn. 56 (1–3) (2004) 89–113.
[3] A. Brandstädt, C. Hundt, Ptolemaic graphs and interval graphs are leaf powers, in: Proc. 8th LATIN, in: LNCS, Springer, 2008 (in press).
[4] A. Brandstädt, V.B. Le, Structure and linear time recognition of 3-leaf powers, Inform. Process. Lett. 98 (4) (2006) 133–138.
[5] A. Brandstädt, V.B. Le, J.P. Spinrad, Graph Classes: A Survey. SIAM Monogr. Discrete Math. Appl. (1999).
[6] A. Brandstädt, V.B. Le, R. Sritharan, Structure and linear time recognition of 4-leaf powers, 2005 (manuscript).
[7] L. Cai, Fixed-parameter tractability of graph modification problems for hereditary properties, Inform. Process. Lett. 58 (1996) 171–176.
[8] M.-S. Chang, M.-T. Ko, The 3-Steiner root problem, in: Proc. 33rd WG, in: LNCS, vol. 4769, Springer, 2007, pp. 109–120.
[9] M.-S. Chang, M.-T. Ko, H.-I. Lu, Linear-time algorithms for tree root problems, in: Proc. 10th SWAT, in: LNCS, vol. 4059, Springer, 2006, pp. 411–422.
[10] M. Charikar, V. Guruswami, A. Wirth, Clustering with qualitative information, J. Comput. Syst. Sci. 71 (3) (2005) 360–383.
[11] Z.-Z. Chen, T. Jiang, G. Lin, Computing phylogenetic roots with bounded degrees and errors, SIAM J. Comput. 32 (4) (2003) 864–879.
[12] Z.-Z. Chen, T. Tsukiji, Computing bounded-degree phylogenetic roots of disconnected graphs, J. Algorithms 59 (2) (2006) 125–148.
[13] M. Dom, J. Guo, F. Hüffner, R. Niedermeier, Error compensation in leaf power problems, Algorithmica 44 (4) (2006) 363–381.
[14] M. Dom, J. Guo, R. Niedermeier, Bounded degree closest k-tree power is NP-complete, in: Proc. 11th COCOON, in: LNCS, vol. 3595, Springer, 2005, pp. 757–766.
[15] R.G. Downey, M.R. Fellows, Parameterized Complexity, Springer, 1999.
[16] P. Erdős, L. Pósa, On the maximal number of disjoint circuits of a graph, Publ. Math. Debrecen 9 (1962) 3–12.
[17] M.R. Fellows, M.A. Langston, F.A. Rosamond, P. Shaw, Efficient parameterized preprocessing for cluster editing, in: Proc. 16th FCT, in: LNCS, vol. 4639, Springer, 2007, pp. 312–321.
[18] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer, 2006.

[19] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, Graph-modeled data clustering: Exact algorithms for clique generation, Theory Comput. Syst. 38 (4) (2005) 373–392.
[20] J. Guo, A more effective linear kernelization for cluster editing, in: Proc. 1st ESCAPE, in: LNCS, vol. 4614, Springer, 2007, pp. 36–47.
[21] T. Jiang, G. Lin, J. Xu, On the closest tree kth root problem, Department of Computer Science, University of Waterloo, 2000 (manuscript).
[22] P.E. Kearney, D.G. Corneil, Tree powers, J. Algorithms 29 (1) (1998) 111–131.
[23] W. Kennedy, G. Lin, 5-th phylogenetic root construction for strictly chordal graphs, in: Proc. 16th ISAAC, in: LNCS, vol. 3827, Springer, 2005, pp. 738–747.
[24] W. Kennedy, G. Lin, G. Yan, Strictly chordal graphs are leaf powers, J. Discrete Algorithms 4 (4) (2006) 511–525.
[25] M. Křivánek, J. Morávek, NP-hard problems in hierarchical-tree clustering, Acta Inform. 23 (3) (1986) 311–323.
[26] L.C. Lau, Bipartite roots of graphs, ACM Trans. Algorithms 2 (2) (2006) 178–208.
[27] L.C. Lau, D.G. Corneil, Recognizing powers of proper interval, split, and chordal graphs, SIAM J. Discrete Math. 18 (1) (2004) 83–102.
[28] G. Lin, P.E. Kearney, T. Jiang, Phylogenetic k-root and Steiner k-root, in: Proc. 11th ISAAC, in: LNCS, vol. 1969, Springer, 2000, pp. 539–551.
[29] Y.-L. Lin, S.S. Skiena, Algorithms for square roots of graphs, SIAM J. Discrete Math. 8 (1) (1995) 99–118.
[30] R. Motwani, M. Sudan, Computing roots of graphs is hard, Discrete Appl. Math. 54 (1) (1994) 81–88.
[31] A. Natanzon, Complexity and approximation of some graph modification problems, Master's Thesis, Department of Computer Science, Tel Aviv University, 1999.
[32] R. Niedermeier, Invitation to Fixed-Parameter Algorithms, Oxford University Press, 2006.
[33] N. Nishimura, P. Ragde, D.M. Thilikos, On graph powers for leaf-labeled trees, J. Algorithms 42 (1) (2002) 69–108.
[34] D. Rautenbach, Some remarks about leaf roots, Discrete Math. 306 (13) (2006) 1456–1461.
[35] R. Shamir, R. Sharan, D. Tsur, Cluster graph modification problems, Discrete Appl. Math. 144 (2004) 173–182.
[36] T. Tsukiji, Z.-Z. Chen, Computing phylogenetic roots with bounded degrees and errors is NP-complete, Theoret. Comput. Sci. 363 (1) (2006) 43–59.
[37] A. van Zuylen, D.P. Williamson, Deterministic algorithms for rank aggregation and other ranking and clustering problems, in: Proc. 5th WAOA, in: LNCS, Springer, 2007 (in press).